

프로그래밍 언어론

변수(Variables)

(Variables)

컴퓨터공학과

이만호



충남대학교
CHUNGNAM NATIONAL UNIVERSITY

변수(Variables)

학 습 목 표

프로그래밍언어에서 사용되는 변수들의 기본적인 속성들에 대해서 학습한다.

학 습 내 용

- •변수와 von Neumann 구조
- •변수의 속성
- •변수의 이름
- •변수의 주소
- •변수의 타입
- •변수의 값



목 차

- 알고가기
- 변수와 von Neumann 구조
- 변수의 속성
- 변수의 이름
- 변수의 주소
- 변수의 타입
- 변수의 값
- 평가하기
- 정리하기



알고가기 1



von Neumann 구조에서 기억장소는 명령형언어(imperative language)의 무엇과 대응할 수 있는가?

- (a) 연산자
- (b) 변수
- (c) 문장
- (d) 함수



알고가기 2



무엇과 무엇을 묶는다는 의미를 가진 동사형 영어 단어로서, 프로그래밍언어론 과목에서 개체(entity)에 속성을 부여한다는 의미로 사용되는 것은?

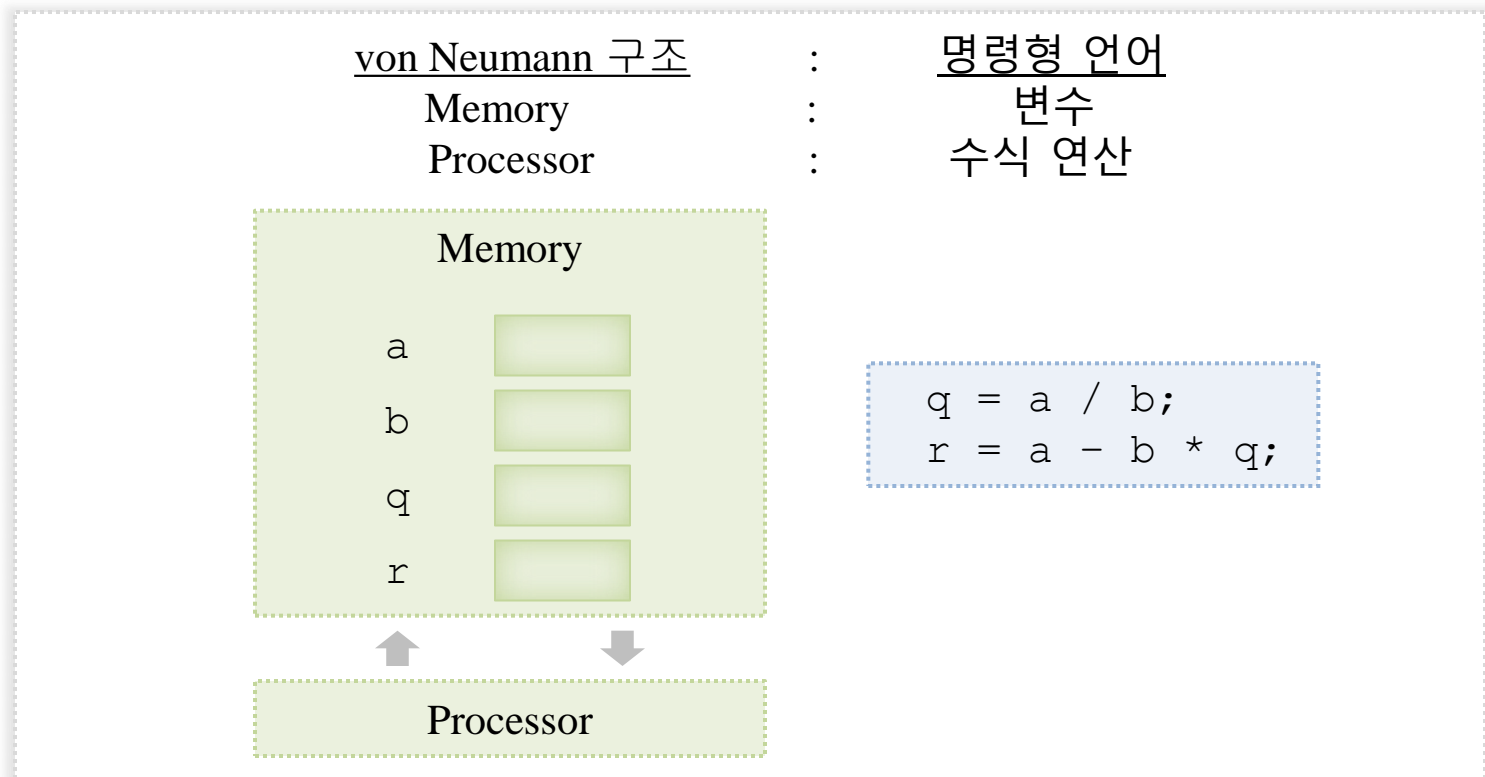
- (a) bind
- (b) chain
- (c) fasten
- (d) tie



| 변수와 von Neumann 구조

명령형언어 (imperative language)는 von Neumann 구조의 컴퓨터를 구현한 형태

- ➔ 변수(variable)는 명령형언어에서 매우 중요한 요소이다.
- ➔ 변수를 사용하지 않고서는 명령형언어로 프로그램을 작성하기가 매우 어렵다.

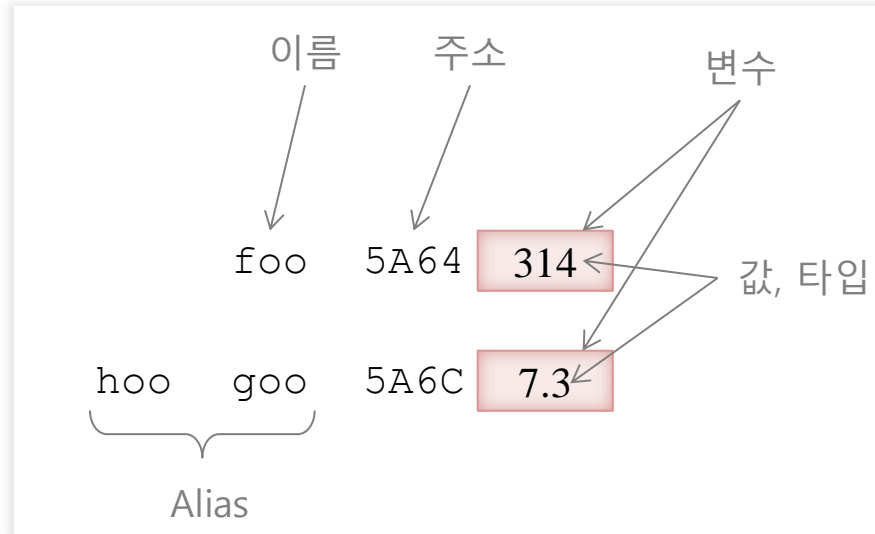


| 변수의 속성(Attributes)

➔ 프로그래밍언어에서 변수(variable)는 von Neumann 구조의 기억장소를 표현한다.

변수의 6가지 속성(Attributes)

- 이름(name)
- 타입(type)
- 주소(address)
- 값(value)
- 영역(scope)
- 존속기간(lifetime)

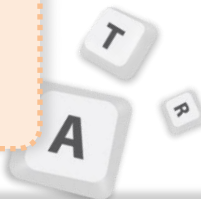


➔ Alias(별칭)

➔ Binding(바인딩)

• 변수는 6가지 속성에 의해서 특징지어진다.

• 변수에 속성을 부여하는 것 ➔ **Binding**



| 변수의 이름(Name) (1)

대부분의 프로그래밍언어에서 변수 이름의 일반적 형태

- ➔ 첫 문자는 영문자
- ➔ 둘 째 문자부터는 영문자나 숫자

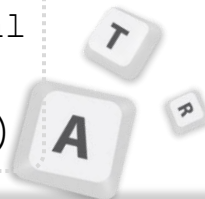
변수 이름과 관련된 주제

➔ 변수 이름의
최대 길이는 몇 자인가?

- 짧으면 readability가 떨어진다.
- Fortran: 6자 → 31자
- Cobol: 30자
- C: 31자
- C#, Ada, Java: 무제한
- C++: 무제한이지만 compiler 구현에 따라 제한이 있을 수 있다.

➔ 변수 이름에
특수 문자 사용이 가능한가?

- Pascal : 사용할 수 없음
- C: `'_'` (예: `average_of_scores`)
- PHP: `$-`
- Perl: `$-`, `@-`, `%-` (변수의 종류에 따라 구별됨)
- Ruby: `@@-:class`, `@-:instance`, `$-:global`
- Fortran : 공백
(예: `Average Of Scores` \equiv `AverageOfScores`)



I 변수의 이름(Name) (2)

변수 이름에서 대문자와 소문자를 구별하는가? (case sensitive?)

→ 대소문자를 구별하는 언어는 readability가 떨어진다.

예 Average, average → 비슷하게 보이지만 다른 변수이다.

→ C, C++, Java: 대소문자 구별(case sensitive)

→ 다른 언어들: 대소문자 구별 없음 (not case sensitive)

특수단어(special word)가 keyword인지, reserved word(예약어)인지?

→ **특수단어** : 프로그램의 구성 요소들을 구별하기 위한 단어

if, for, while, begin, end, int, ...

→ **Keyword** : 특정 문맥에서만 의미를 가지므로, 변수 이름으로 가용가능
→ Readability가 떨어짐

(예: Fortran)

Real	Mean, X, Integer	// 변수 선언문
Integer	Age, N, Real	// 변수 선언문
Integer	= 3.4	
Real	= 345	

→ **Reserved word** : 정의된 목적 이외에 프로그래머 임의로 사용할 수 없음 (예: C)



| 무명 변수

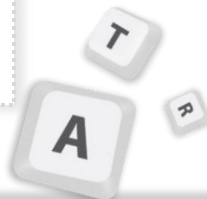
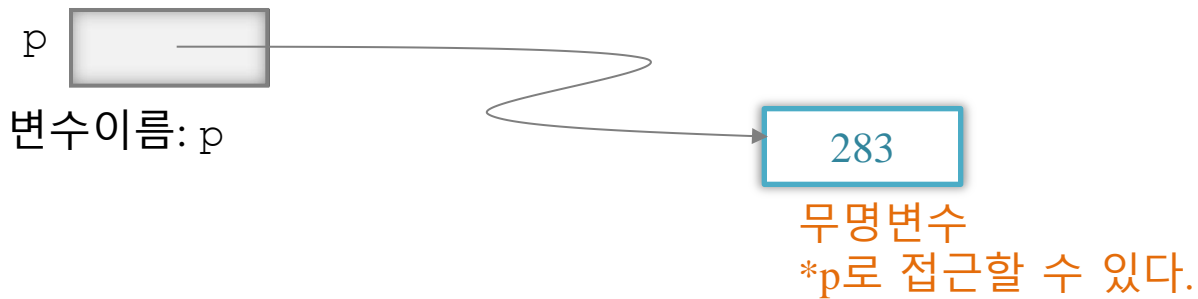
변수는 보통 이름이 주어지지만, 이름이 주어지지 않은 변수도 있을 수 있다.

이름이 없는 변수를 무명변수(anonymous variable)라고 한다.

예

C

```
int *p;  
p = (int *) malloc(sizeof(int));  
*p = 283;
```



| 변수의 주소(Address)

- ➔ 프로그램에서 사용되는 변수는 type 크기만큼의 기억장소가 대응된다.
- ➔ 변수에 대응된 기억장소의 주소. 그 주소 값을 **l-value**라고 한다.
- ➔ 한 변수의 주소는 실행 시점에 따라 다를 수 있다.

```
foo() { int x; ... } // 부프로그램의 내부에 선언된 지역변수
```

- ➔ 동일한 이름의 변수는 프로그램에서 사용된 위치에 따라 주소가 다를 수 있다.

```
int x; // 전역변수
foo() { int x; ... } // foo의 지역변수
goo() { int x; ... } // goo의 지역변수
```

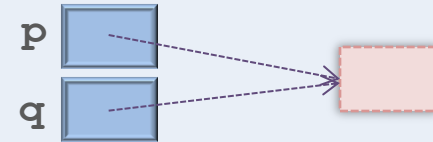
- ➔ 하나의 기억장소에 여러 개의 변수 이름이 대응될 수 있다. (alias 현상)
 - > Alias 현상이 존재하는 프로그램은 readability와 reliability가 좋지 않다.



| Alias 현상이 발생하는 경우

➔ Pointer 사용

```
int    *p, *q;
p = q = (int *)malloc(...);
```



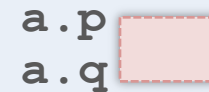
➔ 참조변수(Reference variable) 사용 (C++)

```
int    ans;
int    &ref_ans = ans;
```



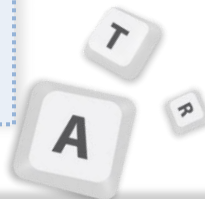
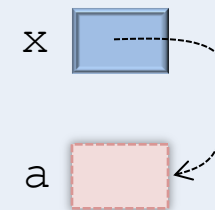
➔ union 사용 (C, C++)

```
union { int    p;
        float q; } a;
```



➔ Call-by-Reference(참조-전달)에 의한 parameter(인자) 전달 (C++)

```
void foo(int &x, ...) { ... }
main() {
    int a;
    ... foo(a, ...) ... }
```



| 변수의 Type

Type의 종류

➔ Primitive Data Type(기본) : 정수형, 실수형, 문자형, 논리형

C가 지원하는 타입 : `int`, `float`, `double`, `char`

➔ Derived Data Type(유도된) : 배열(array), 구조체(record), Pointer, ...

타입으로부터 알 수 있는 정보

➔ 값의 범위(range of values)

➔ 연산의 종류(set of operations)

➔ 유효숫자의 범위(precision) : 실수형의 경우

예-1

정수형으로부터 알 수 있는 정보 (16-bit의 경우)

- 값의 범위: $-32768 \sim 32767$
- 연산의 종류: `+`, `-`, `*`, `/`, `mod`, ...

예-2

Stack class로부터 알 수 있는 정보

- 값의 범위: class 내에 member 데이터로 선언된 변수의 타입에 따라 좌우된다.
- 연산의 종류: class 내에 member 함수로 선언된 `pop`, `push`, ...



| 변수의 값(Value)

변수에 대응되어 있는 기억장소에 저장되어 있는 값

변수와 관련된 값(value)의 종류

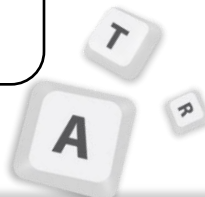
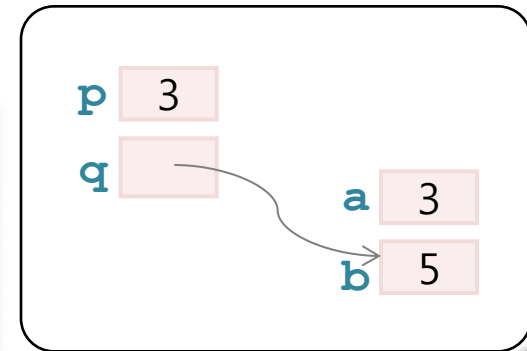
- ➔ 변수의 l-value: 변수의 주소(address)
- ➔ 변수의 r-value: 변수의 값(value)

```
int k;
k = 3;          // k : l-value
k = k + 5;      // k : r-value
```

Parameter passing(인자전달)

- ➔ Call-by-value(값-전달): r-value를 전달
- ➔ Call-by-reference(참조-전달): l-value를 전달

```
int foo(int p, int *q) { ... }
void main()
{   int a=3, b=5;
    ... foo(a, &b)
```



평가하기

마지막으로 내가 얼마나 이해했는지를 한번 확인해 볼까요?
총 4문제가 있습니다.

START



평가하기 1

1. 프로그램에서 사용되는 변수(variable)에 대한 설명으로서 틀리게 설명하고 있는 것은?
- (a) 변수에는 기억장소의 주소가 주어진다.
 - (b) 한 변수의 주소는 실행 시점에 따라 달라질 수 있다.
 - (c) 변수는 이름(name)이 있을 수도 있고 없을 수도 있다.
 - (d) "foo"라는 이름의 변수가 프로그램 내 여러 곳에서 사용되더라도, "foo"라는 이름의 변수에 대한 선언(declaration)은 하나만 있어야 한다.

확인



평가하기 2

2. 다음 중 alias 현상이 발생하지 않는 것은?

- (a) pointer 사용
- (b) union 사용
- (c) Reference variable(참조변수) 사용
- (d) Call-by-value(값-전달) 방법에 의한 parameter passing(인자전달)

확인



평가하기 3

3. Data type의 이름으로부터 얻을 수 있는 정보가 아닌 것은?

- (a) 값(value)의 범위
- (b) 연산(operation)의 종류
- (c) Data type의 변수가 사용하는 기억장소의 주소
- (d) Data type 의 변수가 사용하는 기억장소의 크기



평가하기 4

4. 변수에 할당된 기억장소의 주소를 의미하는 용어는 무엇인가?

- (a) a-value
- (b) l-value
- (c) p-value
- (d) r-value



정리하기

🌱 변수의 6가지 속성

이름(name), Type, 주소(address), 값(value), 영역(scope), 존속기간(lifetime)

🌱 프로그래밍언어 평가 기준에 영향을 끼치는 요인

- > 이름의 일반적 형태: 첫 문자는 영문자이고, 둘째 문자부터는 영문자나 숫자
- > 이름이 주어지지 않는 무명변수(anonymous variable)도 가능하다.

🌱 변수 이름과 주소

- > 동일한 이름의 변수는 프로그램에서 사용된 위치에 따라 주소가 다를 수 있다.
- > 하나의 기억장소에 여러 개의 변수 이름이 대응될 수 있다. (alias 현상)

🌱 변수 이름과 주소

- > 값의 범위(range of values)
- > 연산의 종류(set of operations)